PTO/SB/21 (09-04)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

2154

OIPE JUN 13 2005 PATENT & TRADEMARK OFFICE

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Total Number of Pages in This Submission

Application Number	09/859,660
Filing Date	May 16, 2001
First Named Inventor	Guy Eden
Art Unit	2154
Examiner Name	Ramsey Rafai
Attorney Docket Number	SLA1014

ENCLOSURES ***(check all that apply)***

- ☒ Fee Transmittal Form
- ☐ Fee Attached
- ☐ Amendment / Reply
- ☐ After Final
- ☐ Affidavits/declaration(s)
- ☐ Extension of Time Request
- ☐ Express Abandonment Request
- ☐ Information Disclosure Statement
- ☐ Certified Copy of Priority Document(s)
- ☐ Reply to Missing Parts/ Incomplete Application
- ☐ Reply to Missing Parts under 37 CFR1.52 or 1.53
- ☐ Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition
- ☐ Petition to Convert to a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☐ Request for Refund
- ☐ CD, Number of CD(s) ____
  - ☐ Landscape Table on CD
- ☐ After Allowance Communication to TC
- ☐ Appeal Communication to Board of Appeals and Interferences
- ☒ Appeal Communication to TC **(Appeal Notice, Brief, Reply Brief)**
- ☐ Proprietary Information
- ☐ Status Letter
- ☐ Other Enclosure(s) *(please identify below)*:

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm	Sharp Laboratories of America, Inc.		
Signature			
Printed Name	David C. Ripma, Patent Counsel		
Date	June 10, 2005	Reg. No.	27,672

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below.

Signature			
Typed or printed name	Kimberly Mullen	Date	June 10, 2005

This collection of information is required by 37 CFR 1.5. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/SB/17 (12-04v2)
Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Effective on 12/08/2004.
Fees pursuant to the Consolidated Appropriations Act, 2005 (H.R. 4818).

FEE TRANSMITTAL
For FY 2005

☐ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT	($)	500.00

Complete if Known	
Application Number	09/859,660
Filing Date	May 16, 2001
First Named Inventor	Guy Eden
Examiner Name	Ramsey Rafai
Art Unit	2154
Attorney Docket No.	SLA1014

METHOD OF PAYMENT (check all that apply)

☐ Check ☐ Credit Card ☐ Money Order ☐ None ☐ Other (please identify): _____

☑ Deposit Account Deposit Account Number: 19-1457 Deposit Account Name: Sharp Laboratories

For the above-identified deposit account, the Director is hereby authorized to: (check all that apply)

☑ Charge fee(s) indicated below ☐ Charge fee(s) indicated below, **except for the filing fee**

☑ Charge any additional fee(s) or underpayments of fee(s) under 37 CFR 1.16 and 1.17 ☑ Credit any overpayments

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

FEE CALCULATION

1. BASIC FILING, SEARCH, AND EXAMINATION FEES

	FILING FEES		SEARCH FEES		EXAMINATION FEES		
		Small Entity		Small Entity		Small Entity	
Application Type	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fees Paid ($)
Utility	300	150	500	250	200	100	
Design	200	100	100	50	130	65	
Plant	200	100	300	150	160	80	
Reissue	300	150	500	250	600	300	
Provisional	200	100	0	0	0	0	

2. EXCESS CLAIM FEES

Fee Description	Fee ($)	Small Entity Fee ($)
Each claim over 20 (including Reissues)	50	25
Each independent claim over 3 (including Reissues)	200	100
Multiple dependent claims	360	180

Total Claims	Extra Claims	Fee ($)	Fee Paid ($)
_____ - 20 or HP =	_____	x _____	= _____

HP = highest number of total claims paid for, if greater than 20.

Indep. Claims	Extra Claims	Fee ($)	Fee Paid ($)
_____ - 3 or HP =	_____	x _____	= _____

HP = highest number of independent claims paid for, if greater than 3.

Multiple Dependent Claims	
Fee ($)	Fee Paid ($)
_____	_____

3. APPLICATION SIZE FEE

If the specification and drawings exceed 100 sheets of paper (excluding electronically filed sequence or computer listings under 37 CFR 1.52(e)), the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).

Total Sheets	Extra Sheets	Number of each additional 50 or fraction thereof	Fee ($)	Fee Paid ($)
_____ - 100 =	_____	/ 50 = _____ (round **up** to a whole number)	x _____	= _____

4. OTHER FEE(S)

	Fees Paid ($)
Non-English Specification, $130 fee (no small entity discount)	
Other (e.g., late filing surcharge): Submission of Appeal Brief	500.00

SUBMITTED BY

Signature		Registration No. (Attorney/Agent) 27,672	Telephone 360-834-8754
Name (Print/Type)	David C. Ripma, Reg. No. 27,672		Date June 10, 2005

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re Application of:		)
		)
Inventors:	Guy Eden	)
		)
Serial No.:	09/859,660	) ATTORNEY FILE NO.
		) SLA1014
Filed:	May 16, 2001	)
		) Customer No.: 27518
Title:	SYSTEM AND METHOD FOR DISCOVERING NETWORK COMPONENTS	) Examiner: Ramsey Refai
		)
		) Confirmation No.: 3934
		)
		) Art Unit: 2154

Board of Patent Appeals and Interferences
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

BRIEF ON APPEAL

This is an appeal from the rejection by Examiner Ramsey Refai, Group Art Unit 2154, of claims 1-6, 8-20, and 22-26 as set forth in the CLAIMS APPENDIX, all claims in the application.

REAL PARTY IN INTEREST

The real party in interest is Sharp Laboratories of America, Inc., as assignee of the present application by an Assignment in the United States Patent Office on August 11, 2001, with a recordation date of May 16, 2001 at Reel 011840, Frame 0401.

RELATED APPEALS AND INTERFERENCES

None.

STATUS OF THE CLAIMS

Claims 1-6, 8-20, and 22-26 are in the application.

Claims 1-6, 8-20, and 22-26 are rejected.

Claims 1-6, 8-20, and 22-26 are appealed.

STATUS OF AMENDMENTS

Amendments were made to the claims in an Office Action response received at the PTO on October 20, 2004. The paper of October 20, 2004 was responsive to an initial Office Action mailed September 20, 2004. These claims amendments have been entered, and no further amendments have been submitted.

SUMMARY OF CLAIMED SUBJECT MATTER

The problem addressed by the present invention is presented in the specification at page 1, line 11, through page 3, line 12 (see the specification enclosed in EVIDENCE APPENDIX, ATTACHMENT A). Generally, the problem is associated with network discovery. Upon initialization, a querying device (i.e., a personal computer) conventionally

checks its list of network-connected components (i.e., a printer) by attempting to communicate with every device on the list. Once the list has been checked, the querying device builds a graphical user interface (GUI) to show a user which connected devices are available (see Fig. 1, EVIDENCE APPENDIX, ATTACHMENT B). The problem occurs when a device is no longer connected to the network, or is turned off. Then, the querying device can wait for as long as 30 seconds for a response from a single device. If no response is received (timeout), then the GUI will indicate that the device is not connected (see Fig. 2). However, no GUI is displayed to a user until all the device queries have been resolved.

The Applicant's solution to the problem is simple. Rather than waiting for all the devices to reply, the querying device first builds the GUI, see the timing diagram of Fig. 8. Then, as devices either respond or timeouts occur, the status (availability) of a device is modified. A process for querying network-connected devices to determine availability (claim 1) is described at page 15, ln. 9, through page 16, ln. 12. In its broadest form, Step 1304 builds a GUI that represents device availability. Then, Step 1306 begins querying devices. As described in dependent claims, Step 1305 shows that the devices can initially be represented as unavailable. If a reply is received (Step 1308), the GUI is revised to show the device as available. If a timeout occurs (Step 1312), the device unavailable status is maintained (Step 1314). The device status initially represented in the GUI is arbitrary, since the GUI is updated with actual values once the query process is completed.

A method for building a GUI that represents device availability, independent of system timeouts (claim 13), is described at page 17, ln. 24, through page 18, ln. 9 (see Fig, 14). Step 1402 builds a

GUI of network-connected devices. Step 1404 initially represents devices as unavailable. Step 1406 modifies the GUI to show a device as available in response to receiving a communication from that device. The invention is recited from a systems/device perspective in claim 15, which is described at page 7, ln. 7, through page 8, ln. 2 (see Fig. 3).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Whether claims 1 and 16 are anticipated under 35 U.S.C. 102(b) by the Applicant's admitted prior art (AAPA).

2. Whether claims 2-7, 9-15, and 17-26 are unpatentable under 35 U.S.C. 103(a) with respect to the AAPA, in view of Knodt et al. ("Knodt"; US 5,987,535).

3. Whether claim 8 is unpatentable under U.S.C. 103(a) over the AAPA and Knodt, and further in view of Bahlmann (US 6,393,478).

ARGUMENT

1. The rejection of claims 1 and 16 under 35 U.S.C. 102(b) as anticipated by the Applicant's admitted prior art (AAPA).

In Section 2 of the Final Office Action, claims 1 and 16 have been rejected under 35 U.S.C. 102(b) as anticipated by the Applicant's admitted prior art (AAPA). With respect to claims 1 and 16, the Office Action states that the AAPA describes a step of "at a querying device, building a GUI representing the availability of known network-connected devices; and, querying the known network-connected devices to determine their availability."

"A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987).

Beginning at page 1, ln. 25, the Applicant's specification states that conventional systems "build the GUI to validate device availability only *after* (emphasis added) it has received replies from all the components (network devices) whose existence the application wants to query." This conventional process is described in more detail in the explanation of Fig. 1, where it states that Step 12 sends queries to network-connected devices, and Step 16 waits for all the queries (threads) to return with an answer. Only after these 2 steps, is the GUI built in Step 18 (page 2, ln. 21. through page 3, ln. 4).

Unlike the AAPA, the claimed invention GUI is built *before* queries are sent out to the network-connected devices. In summary, the AAPA builds a GUI after the queries are returned, and the invention of claims 1 and 15 builds the GUI before the queries are sent.

Section A of the Advisory Action of April 29, 2005 states that features upon which the Applicant relies "i.e., GUI is built before queries are sent out" are not recited in the rejected claims. In response, the Applicant notes that claim 1 recites "following the building of the GUI, querying the known network-connected device..." This language clearly differentiates the claimed invention from the AAPA, which builds a GUI only *after* making device queries.

It is the Applicant's opinion that the Examiner is misreading the claim phrase "building a GUI representing the availability..." This phrase should not be read to mean that a GUI is built based upon a query,

upon accurate representations of device availability, or actual availability status. Alternately stated, the phrase "representing the availability... of devices" should not be read as "representing available devices" or "representing devices as available". "Availability" does not mean the same as "available", as there are at least three states of availability: available, not available, and unknown.

However, even if the terms "available" and "availability" are assumed to have an identical meaning, claims 1 and 16 can still be distinguished from the AAPA. If the claims are assumed to recite that a GUI is initially built, where all the devices are marked as available before queries are made, this invention can be distinguished from the AAPA, which builds a GUI only after queries are made. Note, representing a device as available does not necessarily mean that the actual device availability has been determined. The initial representation value, whether it be available or non-available, it arbitrary, since it is replaced with the actual value once the query is completed.

The ***Response to Arguments*** Section of the Final Office Action (page 9, last paragraph) and Section B of the Advisory Action both state that, "the claim language does not explicitly state that the GUI is built without querying the network nor does it state how the GUI checks for availability without querying the network devices..." In response, the Applicant submits that the claims are not attempting to recite a method of "building a GUI without querying the network" or "checking for availability without querying network devices".

The ***Response to Arguments*** Section of the Final Office Action and Section B of the Advisory Action both state that, "the claim language does not show ... how a GUI that represents "the availability of

known network-connected devices" is built without a way to determine what devices are available to display without some type of query." However, this is exactly the novelty of the invention. Devices are initially represented in the GUI, before the query process is begun, and before device availability is actually determined.

Section C of the Advisory Action states that "the features upon which the Applicant relies (i.e., true availability status) are not recited in the rejected claim(s)." The Applicant respectfully notes that the Applicant neither recites true availability status, nor relies upon such a feature to distinguish their invention from the prior art. In fact, the Applicant's position is just the opposite. The prior art shows the "true availability status" of a device. The claimed invention GUI is built before the true device status is known.

Section D of the Advisory Action states that "the features upon which the Applicant relies (i.e., actual availability) are not recited in the rejected claim(s)." Again, the Applicant respectfully notes claim 1 and claim 16 do not recite the feature of "actual availability". In fact, the Applicant's position is just the opposite. The prior art shows the "actual availability" of a device. The claimed invention GUI is built before the actual availability is known.

The AAPA does not describe the building of an availability GUI before querying network-connected devices. Since the AAPA does not describe all the limitations of claims 1 and 15, it cannot anticipate. Claim 16, dependent from claim 15, enjoys the same distinctions from the prior art.

2. The rejection of claims 2-7, 9-15, and 17-26 under 35 U.S.C. 103(a) as unpatentable with respect to the AAPA, in view of Knodt et al. ("Knodt"; US 5,987,535).

Section 6 of the Final Office Action states that claims 2-7, 9-15, and 17-26 have been rejected under 35 U.S.C. 103(a) as unpatentable with respect to the AAPA, in view of Knodt et al. ("Knodt"; US 5,987,535, see EVIDENCE APPENDIX, ATTACHMENT C). With respect to claim 2, the Office Action states that the AAPA fails to teach a method of providing immediate status, but that Knodt provides immediate status indicators, and that it would have been obvious to combine the teachings of the AAPA and Knodt "because Knodt et al's use of immediate status of a user interface in AAPA's method would provide a user the ability to view the status of devices immediately by mimicking machine activities as they occur." With respect to claims 13 and 15, the Office Action states that the AAPA describes all the claim elements.

An invention is unpatentable if the differences between it and the prior art would have been obvious at the time of the invention. As stated in MPEP § 2143, there are three requirements to establish a *prima facie* case of obviousness.

> First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and reasonable expectation of success must both be found in the prior art and not based on applicant's disclosure. *In re Vaeck* 947 F.2d 488, 20 USPQ2d, 1438 (Fed. Cir. 1991).

Knodt states that the object of his invention is to mimic machine activity as it occurs and to provide timing information (col. 2, ln. 10-13). In Fig. 1, Knodt describes a system that includes connected devices such as a display terminal, fax, file server, work station, database system, mail system, and others (col. 3, ln. 7-18). Knodt states that the system could be used as a tutorial, because it shows what is currently happening, what is going to happen, and job access points (col. 3, ln. 62 through col. 4, ln. 3). However, Knodt never discusses how network devices are depicted on the screen before the system determines if the devices are actually accessible. Neither does Knodt describe a screen updating process in the context of network discovery. More explicitly, Knodt does not describe a system that builds a device-availability GUI, prior to actually querying the devices.

As noted in the Background Section of the Applicant's specification (page 2, ln. 8-20), it takes as long as 30 seconds for a time-out to occur, if a device does not respond to a query. Thus, during the initialization process, the ability to mimic machine activities as they occur will not result in an immediate status update. Due to the timeout problem, the ability to mimic machine activity can result in status updates that are significantly delayed. Therefore, it might be said that the AAPA and Knodt methodologies, that mimic machine activities as they occur, teach away from the claimed invention, which is further proof that the claimed invention cannot be considered obvious in light of the prior art.

With respect to the *first prima* facie requirement needed to support a case of obviousness, there is no suggestion to modify the AAPA using Knodt, since Knodt never describes a way of solving the problem

exposed in the AAPA – the building a GUI upon initialization. Since the AAPA exposes the problem of building a GUI upon initialization, and Knodt offers no guidance in this specific area, there can be no motivation to combine reference in a manner that might make the claimed invention obvious.

The ***Response the Arguments***, Section C of the Final Office Action, and Section G of the Advisory Action both state that the AAPA describes a GUI to determine device availability and the Knodt describes a user interface to provide immediate status of imaging devices. The Office Action states that the AAPA teaches the need to validate each component's existence, and the Knodt teaches that it would be desirable to present to a user immediate indications of whether a device is available, or not. The Office Action continues that "one skilled in the art would be motivated to combine the teachings of the AAPA and Knodt because it would create a user interface that can receive an immediate indication to the user of the status of the devices on the network."

However, the issue is not whether a user interface can be created that receives an immediate indication of the status of the devices in the network – this is not even a recited feature of the Applicant's claims. Rather, the issue to be addressed should concern whether there is a motivation to combine the AAPA with Knodt.

The issue of motivation does not concern itself with whether there is some element of commonality between references. If it did, then any two references could be combined merely as the result of a common keyword. Although a prior art device "may be capable of being modified to run the way the apparatus is claimed, there must be a suggestion of motivation in the references to do so." *In re Mills*, 916 F.2d 680, 682, 16

USPQ2d 1430, 1432 (Fed. Cir. 1990). Here, the analysis must determine if there is any motivation to modify the AAPA process of building a GUI only after querying network devices. Knodt may provide a motivation to modify the query process performed by the AAPA. However, the claimed invention is not simply an improved query process.

The motivation to combine should not be based upon the fact the two references are in similar fields of art, or that the two references may be combined, in retrospect, to make some kind of useful invention. Rather, the motivation must be based upon one reference suggesting a modification to a second reference that would make the claimed invention obvious. Here, even if Knodt teaches a user interface that provides immediate device status, Knodt does not teach that a device-availability GUI can be built before device inquiries are made.

The second *prima facie* requirement addresses the same issue from another point of view. Even if an expert were given the two references are a starting point, there is no reasonable expectation that this expert would come up with the claimed invention. The combination of references does not provide an expectation of success that a device-availability GUI can be built without already knowing the accessibility status of networked devices.

With respect to the third requirement to support a *prima facie* case of obviousness, the combination of references does not teach all the limitations of claims 1, 13, and 15. Claims 1, 13, and 15 recite the limitations of a network discovery method that builds a device-availability GUI before sending device availability queries. The AAPA only describes building a GUI after all the device query responses are received. Knodt merely describes a screen that is updated in response the device

query/responses. Further, Knodt does not discuss a network discovery process or initialization process. Thus, the combination of the AAPA and Knodt does not teach the limitation of claims 1, 13, and 15.

The affidavit of Mr. Sridhar Dathathraya (EVIDENCE APPENDIX, ATTACHMENT D) was enclosed in Applicant's response of October 20, 2004, to support the position that the combination of references does not make the claimed invention obvious. The ***Response to Arguments*** Section of the Final Office Action and the Advisory Action both state that the affidavit is insufficient to overcome the rejections made under 35 U.S.C. 102(b) and 103(a) because "the affidavit is only Mr. Dathathraya's opinion and not based upon factual evidence."

As stated in MPEP 716.01(c) III, the opinions of experts in the field have no weight with regard to legal conclusions. However, their opinions are entitled weight with respect for the underlying facts. Further, the Court acknowledges that, "some weight ought to be given to a persuasively supported statement of one skilled in the art on what was not obvious to him." *In re Lindell*, 385 F.2d 456, 155 USPQ 524 (CCPA 1967). In this case Mr. Dathathraya was asked to examine the two prior art references and the claimed invention. Mr. Dathathraya's opinion as an expert in the field was that Knodt did not describe any type of network discovery mechanism. Mr. Dathathraya's conclusion was that, since the AAPA only describes a discovery method that builds a device-availability GUI after sending queries and receiving device availability responses, the combination of references could not suggest a discovery method that built a device-availability GUI prior to sending device availability queries. Of all the involved parties, Mr. Dathathraya's analysis is most likely to be accurate since he is a practicing expert in the field. The Applicant's

respectfully requests that Mr. Dathathraya's technical analysis be given some consideration. If given weight, Mr. Dathathraya's opinions clearly support the underpinnings of the Applicant's argument that a prima *facie case* of obviousness has not been made.

In summary, the combination of the AAPA and Knodt do not explicitly describe all the elements of claims 1, 13, and 15. Neither do the references suggest any modifications that make these claims obvious. Claims 2-6 and 8-12, dependent from claim 1, claim 14, dependent from claim 13, and claims 16-20 and 22-26, dependent from claim 15, enjoy the same distinctions.

3. The rejection of claim 8 under U.S.C. 103(a) as unpatentable over the AAPA and Knodt, and further in view of Bahlmann (US 6,393,478).

In Section 23 of the Final Office Action, claim 8 has been rejected under U.S.C. 103(a) as unpatentable over the AAPA and Knodt, and further in view of Bahlmann (US 6,393,478). The Office Action states that the AAPA and Knodt fail to teach a Sockets connect function, ping function, or NSLookup function. The Office Action states that Bahlmann shows the NSLookup function and that it would have been obvious to combine references "because Knodt's et al's use of building a GUI in real-time and Bahlmann's use of NSLookup function is AAPA's system would allow a user to view instant status information regarding monitored devices by using NSLookup to find the IP address corresponding to the monitored devices and devices to locate the monitoring device."

Bahlmann describes a method for troubleshooting network-connected devices identifiable by a medium access control address (MAC).

A browser is used to find a particular device based upon its MAC, and display device specific data. Update functions are provided to change the data in the internal database, and utility functions are provided to aid in troubleshooting, maintenance, and verification (col. 2, ln. 26-40). However, Bahlmann only provides the conventional method of updating a browser web page in response to device query/responses, and he does not specifically address network discovery processes. Because only conventional query/response mechanisms are described, Bahlmann can be said to teach away from the claimed invention's method of building a GUI before device queries are made, and before device responses are received. As with the AAPA and Knodt, Bahlmann is susceptible to the timeout problem that occurs during initialization when a device does not respond.

Therefore, the combination of the AAPA and Knodt, with Bahlmann, still fails to provide any guidance as to how the GUI building method of the AAPA can be modified. With respect to the base claim (claim 1), from which claim 8 depends, the first *prima facie* requirement to support a case of obviousness has not been met.

With respect to the second *prima facie* requirement, even if the references were combined, there is no expectation of success in the claimed invention from combining the references. Neither Bahlmann nor Knodt provides an expectation that the AAPA GUI building method problem can be modified into one that is not dependent upon the network devices responding to a query.

With respect to the third requirement to support a *prima facie* case of obviousness, the combination of references does not teach all the limitations of claim 1. Claim 1 recites the limitation of building the GUI before sending and receiving device availability queries. The AAPA

only describes building a GUI after all the device query responses are received. Both Knodt and Bahlmann describe screen/browser updates that only occur in response to (after) a device query/response. Thus, the combination of the AAPA, Knodt, and Bahlmann does not teach the limitations of claim 1. Claim 8, dependent from claim 1, enjoys the same distinctions.

SUMMARY AND CONCLUSION

It is submitted that for the reasons pointed out above, the claims in the present application clearly and patentably distinguish over the cited references. Accordingly, the Examiner should be reversed and ordered to pass the case to issue.

Authorization is provide, in the amount of $500.00, to cover the fee for this Appeal Brief. Authorization is given to charge any deficit or credit any excess to Deposit Account No. 19-1457.

Respectfully submitted,

Date: 6/10/05

David Ripma
Registration No. 27,672

Customer Number 27518
David Ripma, Patent Counsel
Sharp Laboratories of America, Inc.
5750 NW Pacific Rim Blvd.
Camas, WA 98607
Telephone: (360) 834-8754
Facsimile: (360) 817-8505
dripma@sharplabs.com

TABLE OF CONTENTS

CLAIMS APPENDIX

1. (Original) In a network of devices, a method for a querying device to determine the availability of network-connected devices, the method comprising:

at a querying device, building a graphical user interface (GUI) representing the availability of known network-connected devices; and

following the building of the GUI, querying the known network-connected devices to determine their availability.

2. (Original) The method of claim 1 further comprising:

at a querying device user interface, issuing a command requesting the availability of devices known to be connected to the network; and

wherein building a GUI representing the availability of known network devices includes building the GUI in real-time, in response to querying device user interface command.

3. (Original) The method of claim 2 further comprising:

following the building of the GUI, representing each of the known network-connected devices in the GUI as unavailable.

4. (Original) The method of claim 3 wherein querying of the known network-connected devices includes spawning a thread from the querying device to query each of the network-connected devices; and

the method further comprising:

receiving a query reply from a network connected device; and

in response to receiving a query reply from a network connected device, changing the GUI representation of that particular network device to available.

5. (Original) The method of claim 4 further comprising:

failing to receive a query reply from a network connected device; and

in response to failing to receive a query reply from a network connected device, maintaining the GUI representation of the particular network device as unavailable.

6. (Original) The method of claim 5 wherein not receiving a query reply from a network connected device includes:

accepting a timeout period for each network connected device query; and

if the timeout period expires before a query reply is received, determining that the particular network connected device is unavailable.

7. Canceled

8. (Original) The method of claim 6 wherein spawning a thread from the querying device to query each of the known network-connected devices includes using a function selected from the group

including a Sockets connect function, a ping function, and a NSLookup function.

9. (Original) The method of claim 6 wherein spawning a thread from the querying device to query each of the known network-connected devices includes requesting a True/False answer;

wherein receiving a query reply from a network connected device includes returning a True answer; and

wherein changing the GUI representation of that particular network device to available includes changing the GUI representation to available in response to a True answer.

10. (Original) The method of claim 9 further comprising:

returning a False answer if the timeout period expires before a query reply is received for a network connected device; and

wherein maintaining the GUI representation of the particular network device as unavailable includes maintaining the GUI as unavailable in response to the False answer.

11. (Original) The method of claim 10 wherein building a graphical user interface (GUI) representing the availability of network includes building a GUI on a computer with a graphical interface; and

wherein issuing commands requesting the availability of the network-connected devices includes requesting the availability of network-connected devices selected from the group including printers, copiers, scanners, faxes, automatic teller machines (ATMs), remote

sensors, virtual private network (VPN) devices, satellite devices, and other computers.

12. (Original) The method of claim 1 further comprising:

accepting a periodic refresh command; and

wherein building a GUI representing the availability of known network-connected devices includes refreshing the GUI in response to a refresh command.

13. (Original) In a network of connected devices, a method of building a graphical user interface (GUI) representing the availability of the network-connected devices independent of system timeouts, the method comprising;

from a querying device, building a graphical user interface (GUI) representing the availability of known network-connected devices;

initially representing the network-connected devices as unavailable; and

modifying the GUI to represent available network devices in response to communicating with those particular network-connected devices.

14. (Original) The method of claim 13 further comprising:

maintaining the GUI to represent unavailable network devices in response to not communicating with those particular network-connected devices.

15. (Original) In a network of connected devices, a system for displaying network device availability, the system comprising:

a querying device having a graphical user interface (GUI) representing the availability of known network-connected devices, the querying device having a network connection port;

at least one device having a network connection port for communications with the querying device; and

wherein the querying device queries known network-connected devices to determine their availability, following the building of the GUI.

16. (Original) The system of claim 15 wherein the querying device has a user interface to accept commands requesting the availability of the network-connected devices; and

wherein the querying device builds a GUI, in real-time, representing the availability of network devices, in response to commands from the querying device user interface.

17. (Original) The system of claim 16 wherein the GUI initially represents each of the network-connected devices as unavailable.

18. (Original) The system of claim 17 wherein the querying device spawns a thread to query each of the network-connected devices, and in response to receiving a query reply from a network connected device, changes the GUI representation of that particular network connected device to available.

19. (Original) The system of claim 18 wherein the querying device maintains the GUI representation of the particular network device as unavailable, in response to not receiving a query reply from that particular network connected device.

20. (Original) The system of claim 19 wherein the querying device further includes an operating system and a timer configured with a default timeout value;

wherein the querying device maintains the GUI representation of the particular network device as unavailable, in response to not receiving a query reply, as follows:

starting the timer at the beginning of each network connected device query; and

if the timeout period expires before a query reply is received from a network connected device, determining that the particular network connected device is unavailable .

21. Canceled

22. (Original) The system of claim 20 wherein the querying device spawns a thread to query each of the network-connected devices by using function selected from the group including a Sockets connect function, a ping function, and a NSLookup function.

23. (Original) The system of claim 22 wherein the querying device GUI requests a True/False answer in response to each network connected device query;

wherein the querying device GUI receives a True answer from available network-connected devices; and

wherein the querying device GUI changes the representation of that particular network device to available in response to a True answer.

24. (Original) The system of claim 23 wherein the querying device generates a False answer in response to a the timeout period expiring before a query reply is received for a network connected device; and

wherein the querying device GUI maintains the representation of the particular network device as unavailable in response to the False answer.

25. (Original) The system of claim 15 wherein the querying device is a computer and the GUI is represented on a visual display attached to the computer; and

wherein the network-connected devices are selected from the group including printers, copiers, scanners, faxes, automatic teller machines (ATMs), remote sensors, virtual private networks (VPNs), satellite devices, and computers.

26. (Original) The system of 20 wherein the timer is configured with a refresh rate value; and

wherein the querying device accepts commands requesting the availability of the network-connected devices at the refresh rate value; and

wherein the querying device refreshes the GUI, in real-time, in response to the refresh rate value.

EVIDENCE APPENDIX

ATTACHMENT A

SYSTEM AND METHOD FOR DISCOVERING AVAILABLE NETWORK COMPONENTS

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention generally relates to computer software and, more particularly, to a system and method of building a graphical user interface (GUI), in real-time, to discover connected network devices.

2. Description of the Related Art

It is conventional for a computer, or a plurality of computers to be networked together for the purposes of cooperation and function sharing. Alternately, a computer or plurality of computers can be linked to devices or elements having specialized functions, such as a printer. The specialized function device can be sometimes be a specialized function computer. It is also conventional that once a network is configured, with links, addresses, and communications established between the network devices, that it remains configured, even when devices are shut down, restarted, or the power recycled. Once a list of components or network devices exists, there is a need to validate each component's existence every time the program is executed. For example, a personal computer (PC) connected to a network of printers will validate communications to each of the network printers when the PC is booted up.

Conventional systems build the GUI to validate device availability only after it has received replies from all the components (network devices) whose existence the application wants to query. This solution is not a real-time operation, as it is characterized by a response time that is relatively slow, often several seconds. This is time that the

system user feels is wasted, as the user is often staring at a display waiting for the GUI to appear. The wait time further depends on the accessibility of the queried devices. If a device is not accessible to the network, it being turned off, broken, or disconnected from the network, the prior art system waits for the expiration of a timeout period, begun at the time the query was initiated. When a device is accessible, the response to the query arrives within approximately 200 milliseconds. When the device is not accessible, the response to the query arrives after the timeout period has expired. The timeout period is not necessarily configurable (i.e., WinSock API). The network operating system may determine the timeout periods. As a result, if only one of the queried network elements is not accessible, the response time is multiplied by a factor of approximately 150, when compared to the case when all the network elements are accessible. This analysis is based on the assumption that a timeout is typically configured to be around 30 seconds, and a query for a network element or online component takes 200 milliseconds. Note, the timeout periods will vary between different operating systems.

Fig. 1 is a flowchart illustrating steps in the method of building a GUI of accessible network devices (prior art). The method starts at Step 10. In Step 12 N threads are spawned from a querying device to each of the N network device. The process must wait for termination of all these threads. In Step 14 all the N spawning threads execute in parallel. In Step 16 the process waits for all the spawned threads to finish and to return their answer. In Step 18, after all the queries are answered, the GUI is built. This is the first time at which the

user can see and interact with the GUI. The GUI gets built according to information (accessible or not accessible) that the N threads have returned.

Fig. 2 is a flowchart illustrating Steps 14 and 16 of Fig. 1 in greater detail (prior art). In Step 14a each thread, here represented by thread 1, performs a query. If the corresponding component is present, the reply to the query will be swift, and the thread will immediately return a positive (True) value, Step 16a. If the corresponding network element is offline, a timeout period will expire (Step 16b), and the query will return a False value (Step 16c).

It would be advantageous if a method existed to more immediately supply a computer user with the results of networked devices accessibility query.

It would be advantageous if a GUI could be built to immediately provide a computer system user of the status of network device accessibility queries.

SUMMARY OF THE INVENTION

The present invention provides an instantaneous real-time indication of all available devices. Given a querying device and a list of devices to which it is connected, a determination is made as to which devices are accessible or available to the querying device through a network. The querying device and the list of devices may or may not be connected to the network. It is assumed that the query is presented to the user through a GUI and that the user is issuing a query command through a keystroke (e.g., enter key) or a mouse click. The solution gives

the user a good experience by delivering the response in real-time, e.g., within less than *0.5* seconds.

Accordingly, a method has been provided for a querying device to determine the availability of known network-connected devices. The method comprises: from a querying device user interface, issuing a command requesting that the availability of the network-connected devices be determined; building a graphical user interface (GUI) in real-time representing the availability of network-connected devices; representing each of the network-connected devices in the GUI as unavailable; and, querying the network-connected devices to determine their availability.

More specifically, the method comprises: spawning a thread from the querying device to query each of the network-connected devices; in response to receiving a query reply from a network connected device, changing the GUI representation of that particular network device to available; or, in response to not receiving a query reply from a network connected device, maintaining the GUI representation of the particular network device as unavailable.

Typically, building a GUI representing the availability of network at a querying device includes building a GUI on a computer with a graphical interface; and, issuing commands requesting the availability of the network-connected devices includes requesting the availability of network-connected devices selected from the group including printers, copiers, scanners, faxes, computers, and equivalent devices.

Additional details of the above-mentioned method, and a system for querying the availability of network of connected devices are provided below.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a flowchart illustrating steps in the method of building a GUI of accessible network devices (prior art).

Fig. 2 is a flowchart illustrating Steps 14 and 16 of Fig. 1 in greater detail (prior art).

Fig. 3 is a schematic block diagram illustrating the present invention system for displaying network device availability, in a network of connected devices.

Fig. 4 is a sample illustration of the GUI from Fig. 3.

Fig. 5 is another sample illustration of the GUI from Fig. 3, following the return of the availability queries.

Fig. 6 is a flowchart illustrating steps in the present invention method of building a GUI in real-time.

Fig. 7 is a detailed illustration of Steps 608 and 610 of Fig. 6.

Fig. 8 is a timing diagram illustrating the above-described present invention method.

Fig. 9 depicts sample code that represents the known network-connected devices as unavailable when the GUI is initialized.

Fig. 10 is sample code depicting the thread spawning function.

Fig. 11 is sample code depicting the attempt to establish a socket connection.

Fig. 12 illustrates the operation of the connect() function.

Fig. 13 is a flowchart illustrating a method for a querying device to determine the availability of network devices in a network of connected devices.

Fig. 14 is an alternate representation of the method of building a graphical user interface (GUI) representing the availability of network-connected devices independent of system timeouts.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Fig. 3 is a schematic block diagram illustrating the present invention system for displaying network device availability, in a network of connected devices 100. A querying device 102 has a graphical user interface (GUI) 104 representing the availability of known network-connected devices. The known network-connected devices shown are: device 1 (106), device 2 (108), device 3 (110), device 4 (112), device 5 (114), and device 6 (116). Although six network-connected devices are shown it should be understood that the invention is not limited to any particular number of network-connected devices. It should also be understood that although only a single querying device is shown, the invention is not necessarily so limited. In some aspects of the invention, the querying device 102 is a computer and the GUI 104 is represented on a visual display attached to the computer 102. The network-connected devices 106-116 are selected from the group including printers, copiers, scanners, faxes, automatic teller machines (ATMs), remote sensors, virtual private network (VPN) elements, satellite elements, computers, and equivalent

devices. There are devices with many other functions that could also be mentioned as being connected to a network.

The querying device 102 has a network connection port connected to line or network connection 118. At least one device (six are shown) has a network connection port for communications with the querying device 102 on line 118. The querying device 102 has, or is connected to a user interface, such as a mouse or keyboard to accept commands requesting the availability of the network-connected devices 106-116. In some aspects of the invention the request is embedded in software and automatically enabled in response to an event such as powering up the querying device 102. The querying device 102 builds the GUI 104, in real-time, representing the availability of network devices 106-116, in response to commands from the querying device user interface 118.

In some aspects of the invention, the real-time building of the GUI 104 occurs within approximately 0.5 seconds of the command from the user interface 118, or in response to a refresh command. The exact time may vary in response to devices, operating systems, and network connections. The term real-time is intended to refer to a very brief period of time that the querying device user perceives to be for instantaneous or almost instantaneous. Following the building of the GUI 104, the querying device 102 queries the known network-connected devices 106-116 to determine their availability.

Fig. 4 is a sample illustration of the GUI 104 from Fig. 3. In this example, device 1 (106) is Cougar01, device 2 (108) is Cougar02, device 3 (110) is Leopard01, device 4 (112) is Leopard02, device 5 (114) is

Leopard03, and device 6 (116) is Leopard04. The querying device GUI 104 initially represents each of the known network-connected devices 106-116 in the GUI as unavailable. Each of the network-connected devices 106-116 is represented by an icon when available. If the devices are copiers, the icons can be made to resemble a copier. If the devices are not available, the unavailable state can be represented as a "crossed-out" icon, or an "X" superimposed over the icon. There are a number of different ways in which available and unavailable devices can be represented and the invention is not limited to the representations of the example in Fig. 4.

Returning to Fig. 3, the querying device 102 spawns a thread to query each of the network-connected devices 106-116, and in response to receiving a query reply from a network connected device, changes the GUI 104 representation of that particular network connected device to available. The querying device 102 maintains the GUI representation of the particular network device as unavailable, in response to not receiving a query reply from that particular network connected device. More particularly, the querying device 102 further includes an operating system (not shown) and a timer 120 configured with a default timeout value. In some aspects of the invention the operating system provides the default timeout value. In other aspects, the user is able to configure the timeout value for the present invention availability GUI. In either case, or regardless of the default timeout value, the GUI 104 is built instantaneously and the available devices are updated in real-time.

The querying device 102 maintains the GUI representation of the particular network device as unavailable, in response to not receiving a query reply, as follows. The timer 120 is started at the beginning of

each network connected device query. If the timeout period expires before a query reply is received from a network-connected device, that the particular network connected device is determined to be unavailable.

In some aspects of the invention the timer 120 is configured with a refresh rate value. Then, the querying device 102 accepts commands requesting the availability of the network-connected devices at the refresh rate value. The GUI 104 is refreshed, in real-time, in response to the refresh rate value. That is, queries are made again, and the GUI changes in response to the queries as described above. In some aspects the refreshing GUI again assumes that all devices are initially unavailable, and the GUI changes state to represent communicating devices as available in real-time. Alternately, the GUI is initiated using the GUI status from the previous refresh cycle. For example, the GUI may be refreshed every 60 seconds. Frequent refresh rates are not a penalty, since the GUI refresh process does not hang the system up in waiting for unavailable network-connected device timeouts.

The querying device GUI 104 requests a True/False answer in response to each network connected device query. The querying device GUI 104 receives a True answer from available network-connected devices, and changes the representation of that particular network device to available in response to a True answer. Likewise, the querying device 102 generates a False answer in response to a the timeout period expiring before a query reply is received for a network connected device, and the querying device GUI 104 maintains the representation of the particular network device as unavailable in response to the False answer.

Fig. 5 is another sample illustration of the GUI 104 from Fig. 3, following the return of the availability queries. Out of a total of six copiers, four are active. The GUI changes the icon for each of these available copiers, from unavailable (initial state) to available, relatively quickly. The two copiers that are unavailable maintain the unavailable icon that was initially set up when the GUI was first built.

Returning to Fig. 3, in some aspects of the invention the querying device 102 spawns a thread to query each of the network-connected devices 106-116 by using a Sockets connect function to attempt a socket connection to each of the network-connected devices.

The present invention improves upon prior art solutions in two aspects:

1. When N devices are queried in the present invention, the average, minimum and maximum response time, from query initiation to GUI presentation is immediate, or o(1), while the prior art response time is o(<timeout-period>), or dependent upon externally controlled factors that make the response time lengthy;

2. When k devices are not accessible, the present invention response time is again immediate, or o(1), while the prior art response time is again lengthy, or o(<timeout-period>).

The benefit of the invention results from the real-time GUI response. There is a list of components (devices) in the network whose existence needs to be validated. The algorithm attempts to open a socket connection in order to verify whether or not the remote component is alive, using the Sockets connect function for example. While prior art methods also use socket connections to discover network-connected devices, the

present invention utilizes socket connections in a new combination, executed in parallel with a GUI context.

The invention builds a GUI depicting all the components as disabled. Subsequently, it spawns N threads, one per each component in the querying device's list. Every thread queries the corresponding device and returns a True/False answer. If the device is alive, the query returns immediately with a True answer and enables the corresponding GUI element in the querying device, by showing the device as being available. If the queried device is offline, the query returns False, but only after a timeout period. A False reply indicates that the device is offline and instructs the querying thread not to change the state of the GUI, leaving the icon disabled (shown as unavailable).

Fig. 6 is a flowchart illustrating steps in the present invention method of building a GUI in real-time. The method begins at Step 600. At Step 602 the GUI is built. The application constructs its GUI representing every known network component (device) with a corresponding GUI icon or representation. In Step 604 the GUI represents every GUI device with its 'disabled' or unavailable state. In Step 606 N threads are spawned. The process does not wait for termination of those threads. In Step 608 all the threads execute in parallel. In Fig. 610 the GUI is modified to depict available devices.

Fig. 7 is a detailed illustration of Steps 608 and 610 of Fig. 6. In Step 608a each thread, here represented by thread 1, performs a query. If the corresponding component (device) is present, the reply to the query will be swift, and the thread will immediately return a positive (True) value, Step 608b. The thread will immediately replace the unavailable

icon with an available icon, and the thread will terminate. That is, the GUI is modified in response to the True answer (Step 610). If the corresponding network element is offline, a timeout period will expire (Step 608c), and the query will return a False value (Step 608d). In response to a False answer the GUI device status is maintained as unavailable.

Fig. 8 is a timing diagram illustrating the above-described present invention method. In this figure, threads 2, i, and N-1 timeout. The remaining threads return a True response, and the GUI changes to show these threads (network-connected devices) as available.

Some functions, such as connect(), will timeout automatically. The timeout for connect() affects non-blocking as well as blocking operations. The GUI application does not have any control over the timeout period for these functions, however, the network system alone determines when their timeout occurs. These network-system timeouts are related to the timeouts implemented for the protocols in use (e.g., ARP timeout, TCP SYN, ACK timeouts, or DNS query timeouts). The WinSock API does not provide a way to detect or change these network-system timeout values.

Fig. 9 depicts sample code that represents the known network-connected devices as unavailable when the GUI is initialized. This function gathers an array of known GUI components (devices). The function first disables all the GUI components, to let them appear offline (See Fig. 4), and then starts a thread per component that will validate the component's existence, and enable the components that are online.

Fig. 10 is sample code depicting the thread spawning

function. Thread is spawned, one per network component (device). The thread queries to determine if the component is alive. If it's alive, the QueryRemoteHost() function will return immediately and the thread will enable (show as available) the GUI icon or representation corresponding to the network device. If, however, the network device is offline and does not respond to the socket connection, the function call OueryRemoteHost() returns after a timeout period, and the thread terminates, not changing the GUI.

Fig. 11 is sample code depicting the attempt to establish a socket connection. This function gets an IP address as input and attempts to establish a socket connection with the remote host. If the component is alive, the function will return immediately with a positive return value (True), but if the network component is not alive, then the function is time extensive and will return only upon timeout.

Since a stream (TCP) client is connection-oriented, it must initiate a connection to create an association. This is done by calling the connect() function, which initiates the creation of a virtual circuit on a TCP socket, or sets a default socket name for a UDP socket. For example:

```
int PASCAL FAR connect (SOCKET s, / * an. unconnected
socket */
struct sockaddr FAR addr,      /*  remote port and I P addr */
int namelen) ;                 /* addr structure length */
```

S socket handle

addr: pointer to a socket address structure (always a sockaddr_in structure for TCP/IP)

l2amelefl: length of structure pointed to by *addr*

The connect() function returns zero on success or SOCKET_ERROR on failure. For a TCP socket, the most common error is usually WSAECONNREFUSED (10061). There are only a few cases that cause this error: The server is not running, the *sin port* is incorrectly initialized on the client (or server), or the wrong IP address is selected.

Fig. 12 illustrates the operation of the connect() function. The connect() function assigns the remote IP address, port Number, and implicitly names the local socket, if not yet explicitly named. It also initiates communication to the server socket over the network.

The invention could be implemented in any given programming language, such as Java, Basic, etc. The invention can use any protocol to discover a network component, such as ping, NSLookup, etc. If needed, the invention can be called within a timer procedure. In that case, the GUI is updated periodically.

Fig. 13 is a flowchart illustrating a method for a querying device to determine the availability of network devices in a network of connected devices. Although the method is depicted as a sequence of numbered steps for clarity, no order should be inferred from the numbering unless explicitly stated. The method begins at Step 1300. Step 1302, at a querying device user interface, issues a command requesting the availability of devices known to be connected to the network. Step 1304 builds a GUI representing the availability of known network-connected devices. Step 1306, following the building of the GUI, queries the network-connected devices to determine their availability to

the querying device. Building a GUI representing the availability of known network devices in Step 1304 includes building the GUI in real-time, in response to querying device user interface command. Building the GUI in real-time includes building the GUI within 0.5 approximately seconds of the query device user interface command. Alternately, the real-time response can be considered as one that appears instantaneous, or almost instantaneous to the user.

Step 1305, following the building of the GUI, represents each of the known network-connected devices in the GUI as unavailable. Querying the known network-connected devices in Step 1306 includes spawning a thread from the querying device to query each of the network-connected devices. Then, Step 1308 receives a query reply from a network connected device. Step 1310, in response to receiving a query reply from a network connected device, changes the GUI representation of that particular network device to available. Likewise, Step 1312, fails to receive a query reply from a network connected device. Step 1314, in response to failing to receive a query reply from a network connected device, maintains the GUI representation of the particular network device as unavailable.

In some aspects of the invention, failing to receive a query reply (Step 1312) includes substeps. Step 1312a accepts a timeout period for each network connected device query. Step 1312b, if the timeout period expires before a query reply is received, determines that the particular network connected device is unavailable.

In some aspects of the invention, spawning a thread from the querying device to query each of the known network-connected devices in

Step 1306 includes using a function selected from the group including a Sockets connect function, a ping function, and a NSLookup function.

In some aspects of the invention, spawning a thread from the querying device to query each of the known network-connected devices in Step 1306 includes requesting a True/False answer. Then, receiving a query reply from a network connected device in Step 1308 includes returning a True answer. Changing the GUI representation of that particular network device to available in Step 1310 includes changing the GUI representation to available in response to a True answer.

Step 1312c returns a False answer if the time-out period expires before a query reply is received for a network connected device. Then, maintaining the GUI representation of the particular network device as unavailable in Step 1314 includes maintaining the GUI as unavailable in response to the False answer.

In some aspects of the invention, building a graphical user interface (GUI) representing the availability of network in Step 1304 includes building a GUI on a computer with a graphical interface. Issuing commands requesting the availability of the network-connected devices in Step 1302 includes requesting the availability of network-connected devices selected from the group including printers, copiers, scanners, faxes, automatic teller machines (ATMs), remote sensors, VPN devices, satellite devices, other computers, and equivalent devices.

In some aspects of the invention a further step, Step 1316 accepts a periodic refresh command. Then, the method returns to Step 1304 where the GUI representing the availability of known network-

connected devices is rebuilt or refreshed in response to the refresh command of Step 1316.

Fig. 14 is an alternate representation of the method of building a graphical user interface (GUI) representing the availability of network-connected devices independent of system timeouts. The method starts at Step 1400. Step 1402, from a querying device, builds a GUI representing the availability of known network-connected devices. Step 1404 initially represents the network-connected devices as unavailable. Step 1406 modifies the GUI to represent available network devices in response to communicating with those particular network-connected devices. Step 1408 maintains the GUI to represent unavailable network devices in response the not communicating with those particular network-connected devices.

Examples of a system and method of providing a real-time GUI to depict the availability of known network-connected devices have been described above. The examples were intended to be as independent of particular operating systems, protocols, and coding languages as possible. Embodiments of the invention in specific operating systems, protocols, and languages will occur to those skilled in the art.

WE CLAIM:

ATTACHMENT B

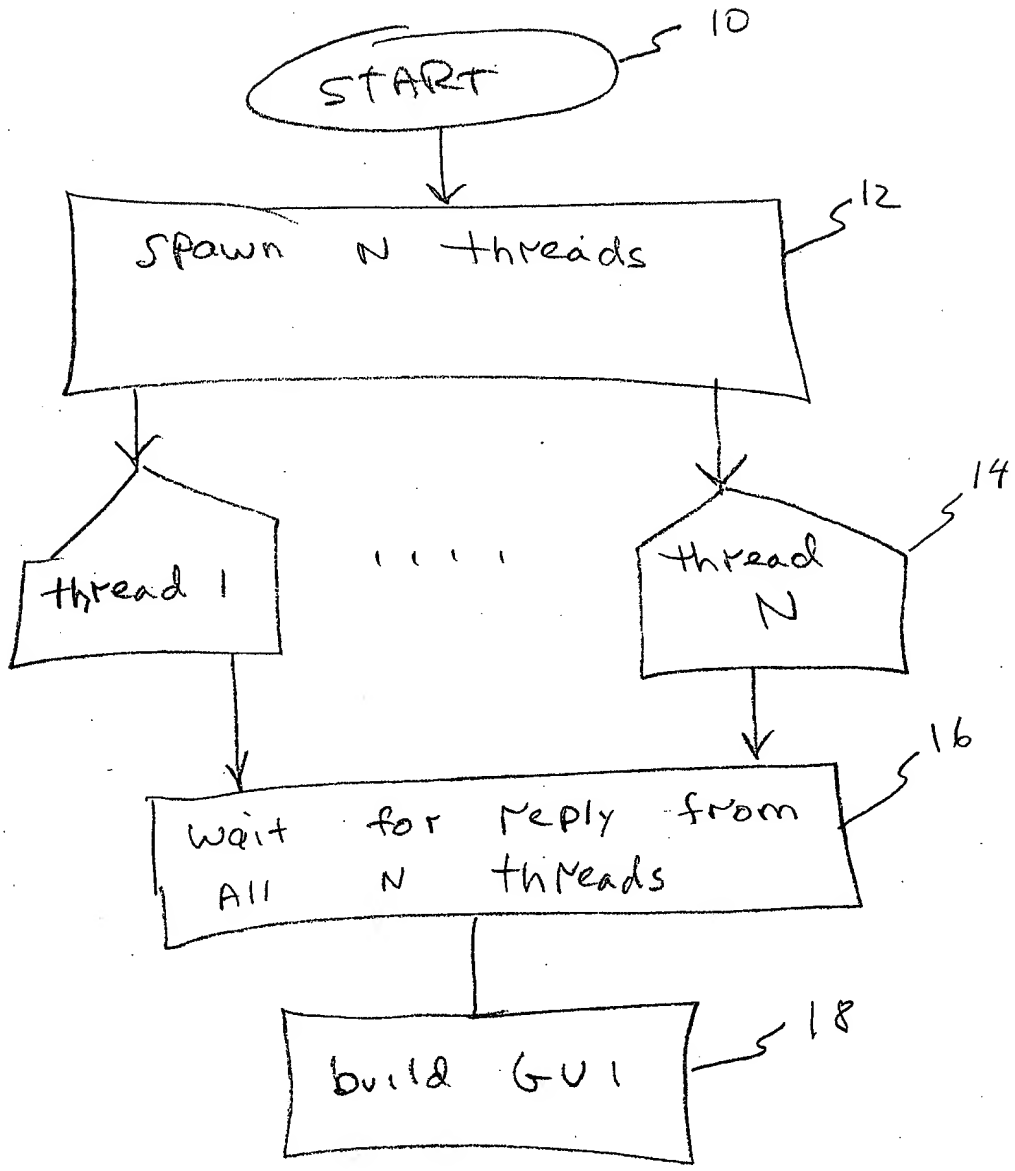


Fig. 1 (prior art)

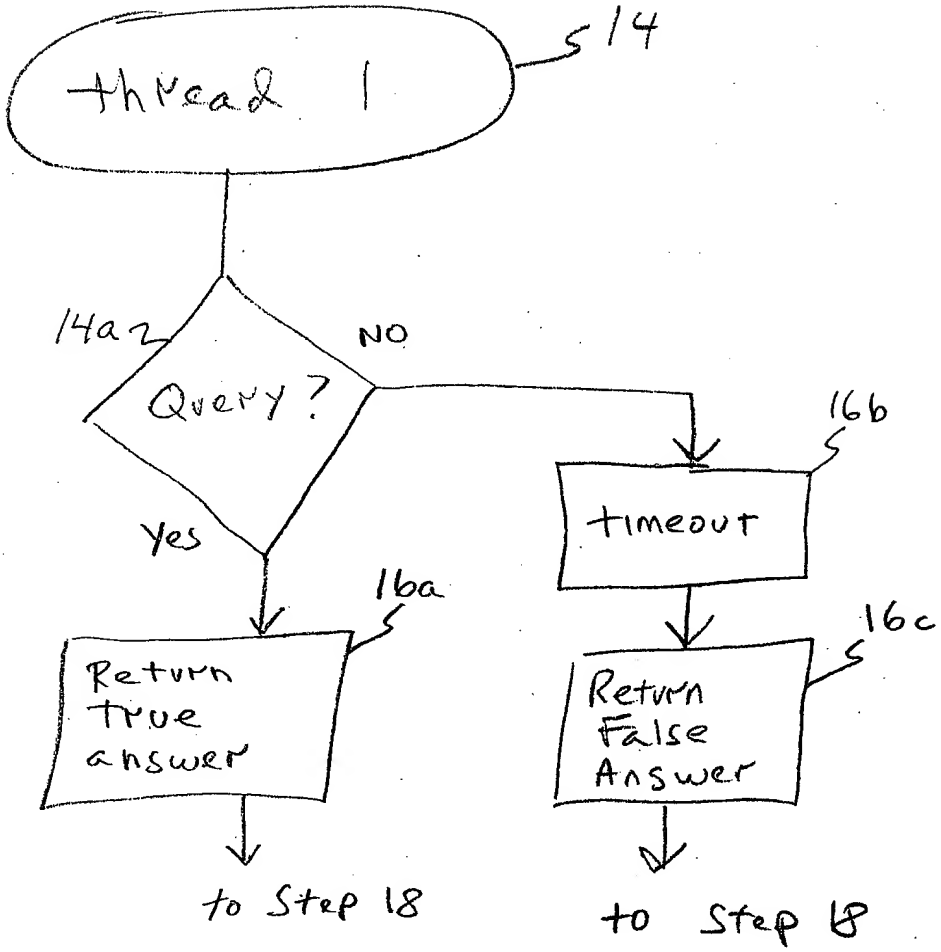


Fig. 2 (prior art)

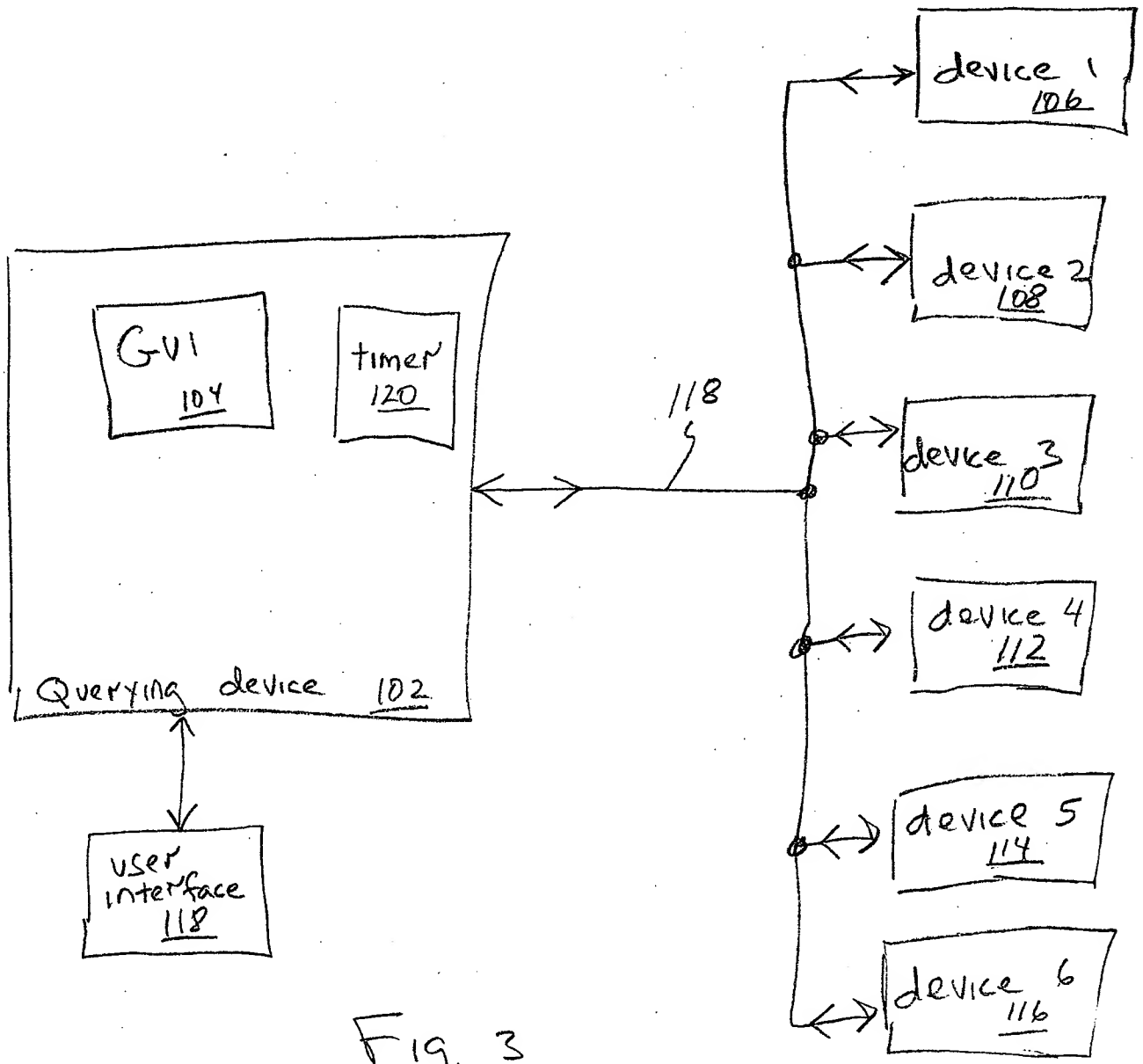

device 1
106
device 2
108
device 3
110
device 4
112
device 5
114
device 6
116
GUI
104
timer
120
118
Querying device 102
user interface
118

Fig. 3

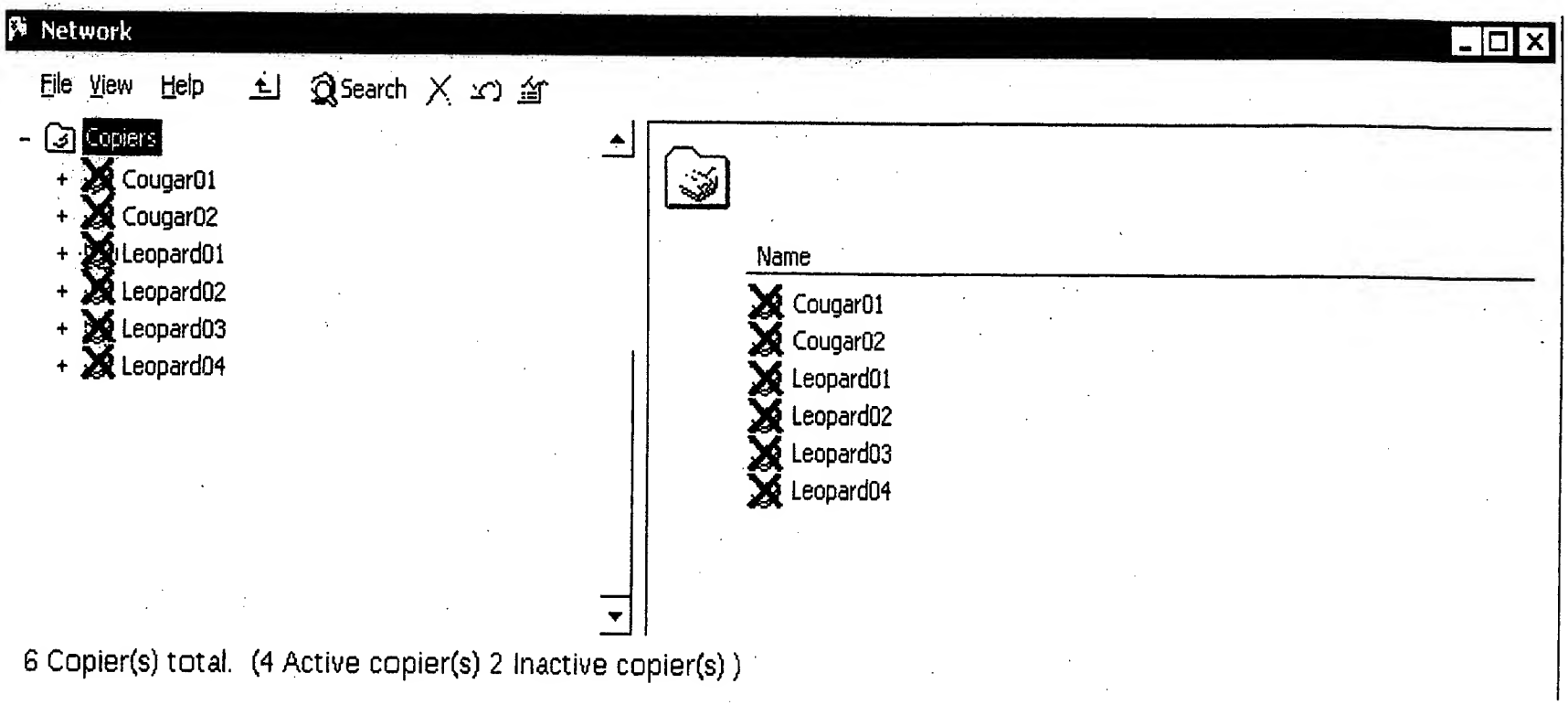

Network
File View Help
Search
Copiers
Cougar01
Cougar02
Leopard01
Leopard02
Leopard03
Leopard04
Name
Cougar01
Cougar02
Leopard01
Leopard02
Leopard03
Leopard04
6 Copier(s) total. (4 Active copier(s) 2 Inactive copier(s))

Fig. 4

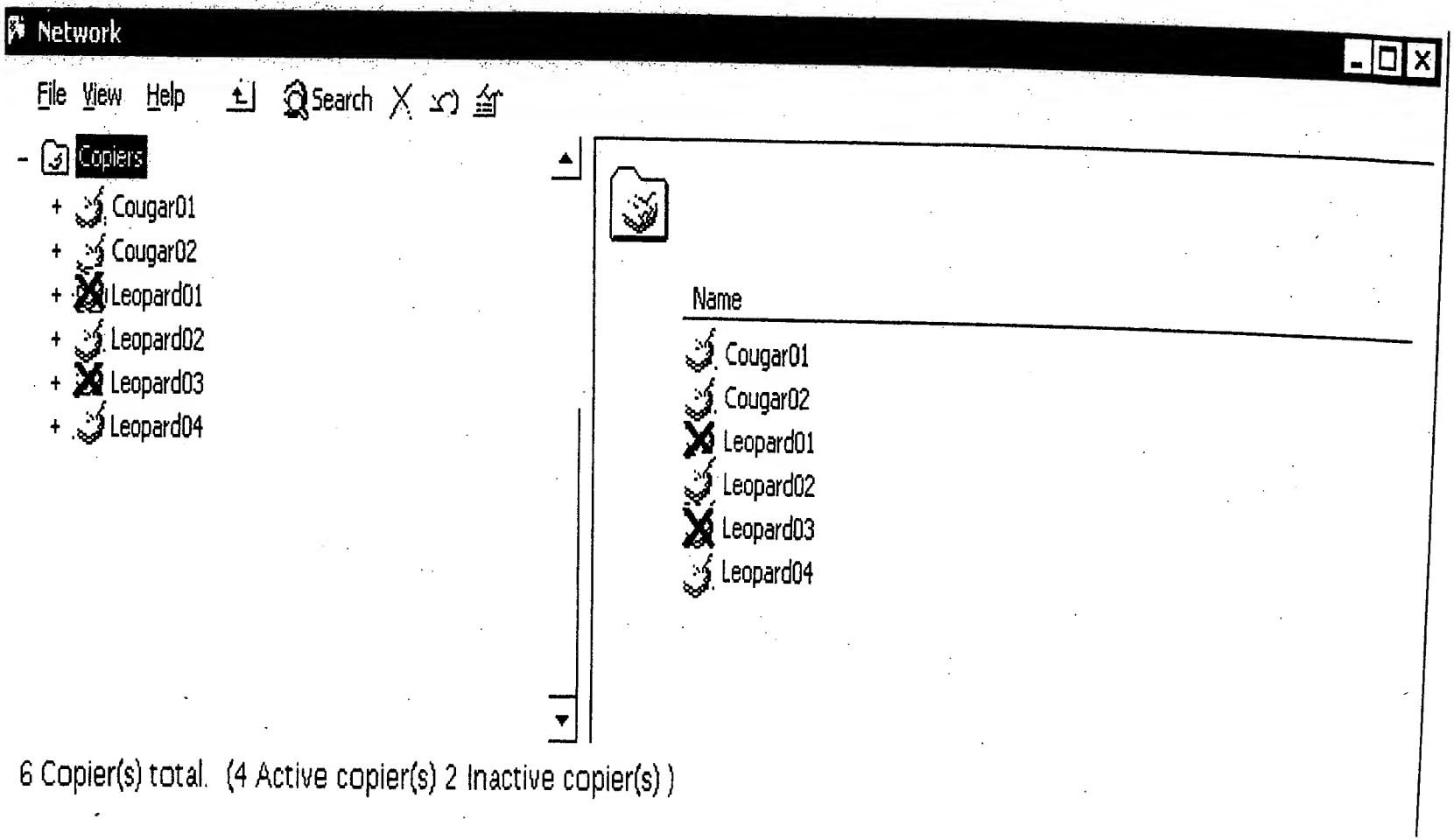

Network
File View Help
Search
Copiers
Cougar01
Cougar02
Leopard01
Leopard02
Leopard03
Leopard04
Name
Cougar01
Cougar02
Leopard01
Leopard02
Leopard03
Leopard04
6 Copier(s) total. (4 Active copier(s) 2 Inactive copier(s))

Fig. 5

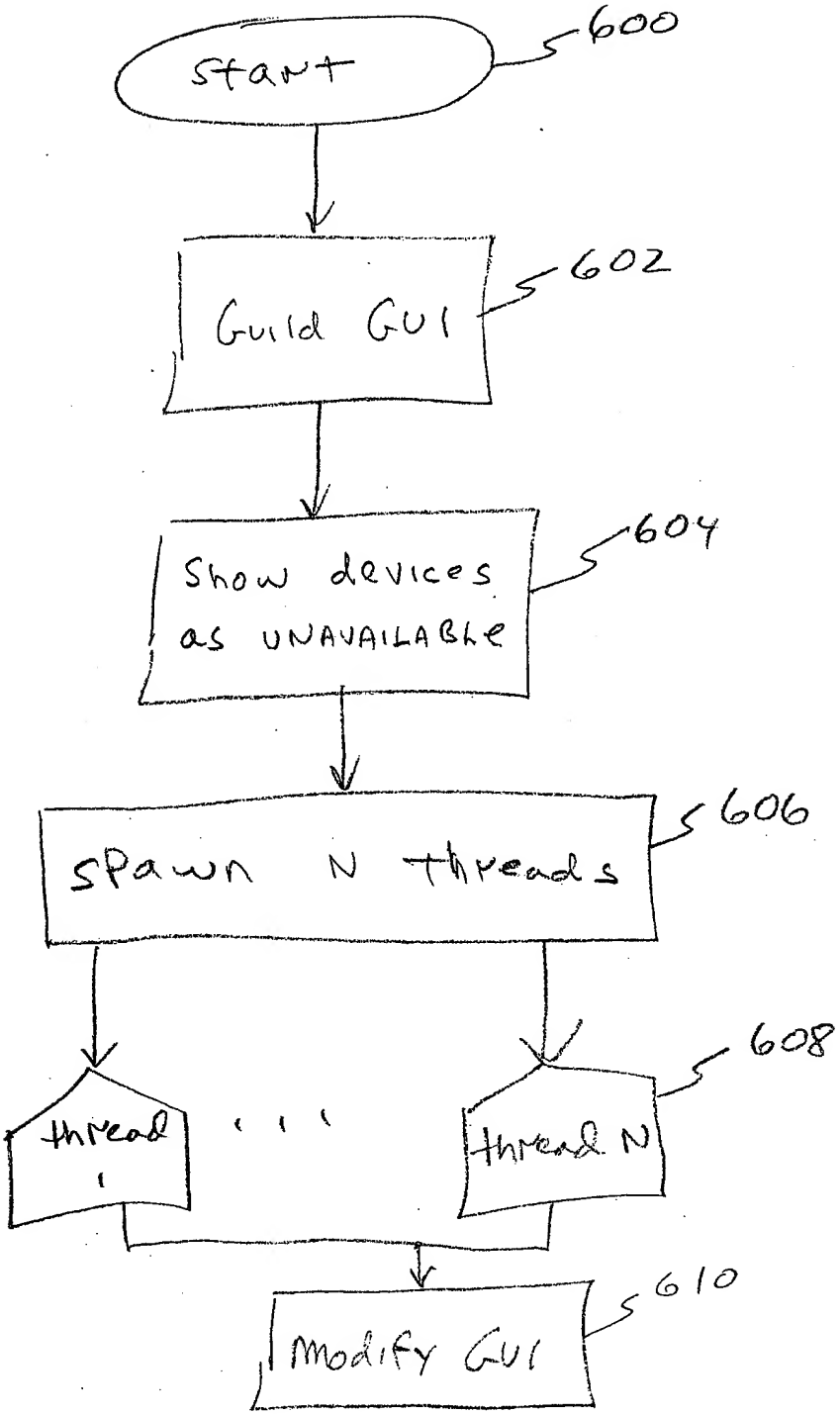

start
600
Build GUI
602
Show devices as UNAVAILABLE
604
spawn N threads
606
thread 1
, , ,
thread N
608
Modify GUI
610

Fig. 6

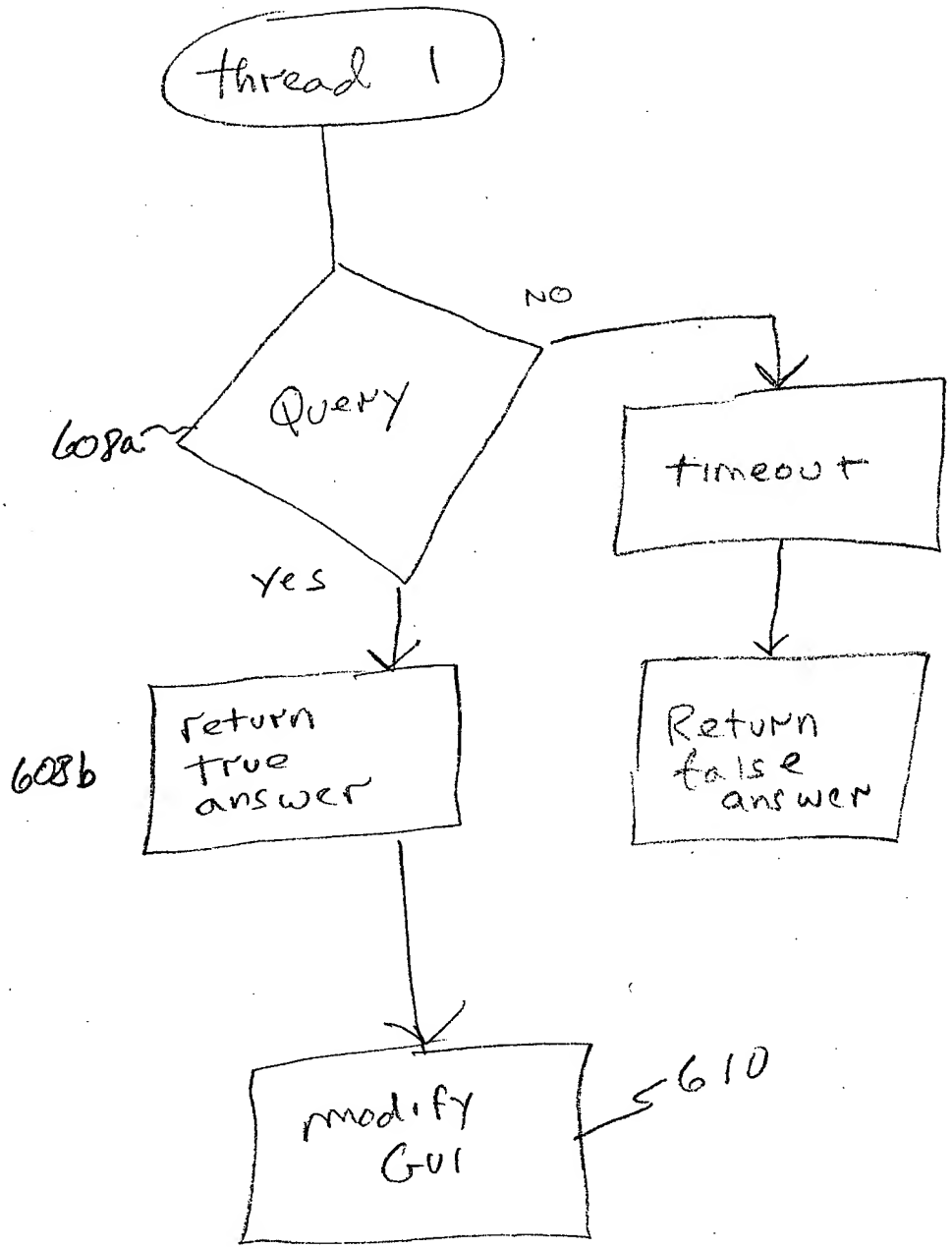

thread 1
Query
608a
NO
timeout
yes
608b
return true answer
Return false answer
modify Gui
610

Fig. 7

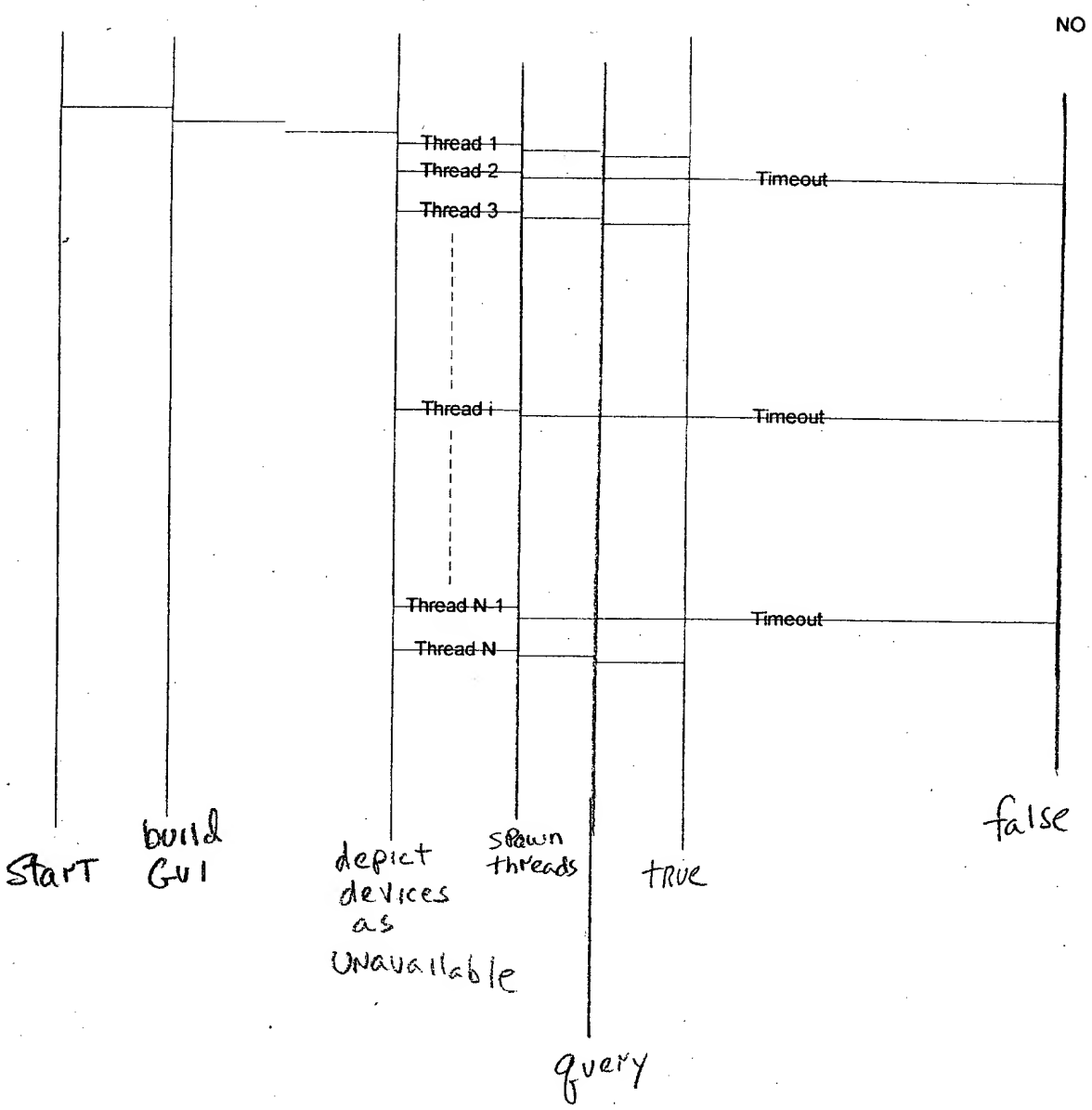

NO
Thread 1
Thread 2
Timeout
Thread 3
Thread i
Timeout
Thread N-1
Timeout
Thread N
Start
build GUI
depict devices as Unavailable
spawn threads
query
true
false

Fig. 8

Code:

```
// DoEnable() gets an array of GUI components, and will enable/disable the components based on their availability in the network.
void DoEnable(GUIComponent arGUI[], unsigned short  sz)
{
        // Disable entire GUI
        for (unsigned short I=0;I<sz;I++)
                arGUI[I].DisableComponent ();

        for (I=0;I<sz;I++)
        {
                AfxBeginThread(ValidateRemoteHost, (LPVOID) &arGUI[I]);
        }
}
```

Fig. 9

```
struct GUIComponent
{
        // will return IP address associated with the network component that the current GUI element represents.
        unsigned long GetIPAddress ();
        void EnableComponent(); // display component as active
        void DisableComponent(); // display component as inactive.
};

// worker thread which will attempt a socket connection with remote host.
UINT ValidateRemoteHost( LPVOID pParam )
{
        GUIComponent * pGUIComponent =( GUIComponent *)pParam;

        if (QueryRemoteHost(pGUIComponent->GetIPAddress()) )
                pGUIComponent->EnableComponent();
        return 0;
}
```

Fig. 10

```
BOOL QueryRemoteHost(unsigned long lIpAdd)
{
        BOOL bRetVal=FALSE;
        // create new socket
        SOCKET S = socket( AF_INET, SOCK_STREAM, IPPROTO_IP);
        if ( INVALID_SOCKET == S)
                return FALSE;

        SOCKADDR_IN dest_sin;
        ZeroMemory(&dest_sin, sizeof(SOCKADDR_IN));
        dest_sin.sin_family = AF_INET;
        dest_sin.sin_port = htons(21);
        dest_sin.sin_addr.S_un.S_addr = lIpAdd;

        if (SOCKET_ERROR == connect(S,(PSOCKADDR)&dest_sin, sizeof( dest_sin)))
                bRetVal = FALSE;
        else
                bRetVal = TRUE;

        closesocket(S);
        return bRetVal;
}
```

Fig. 11

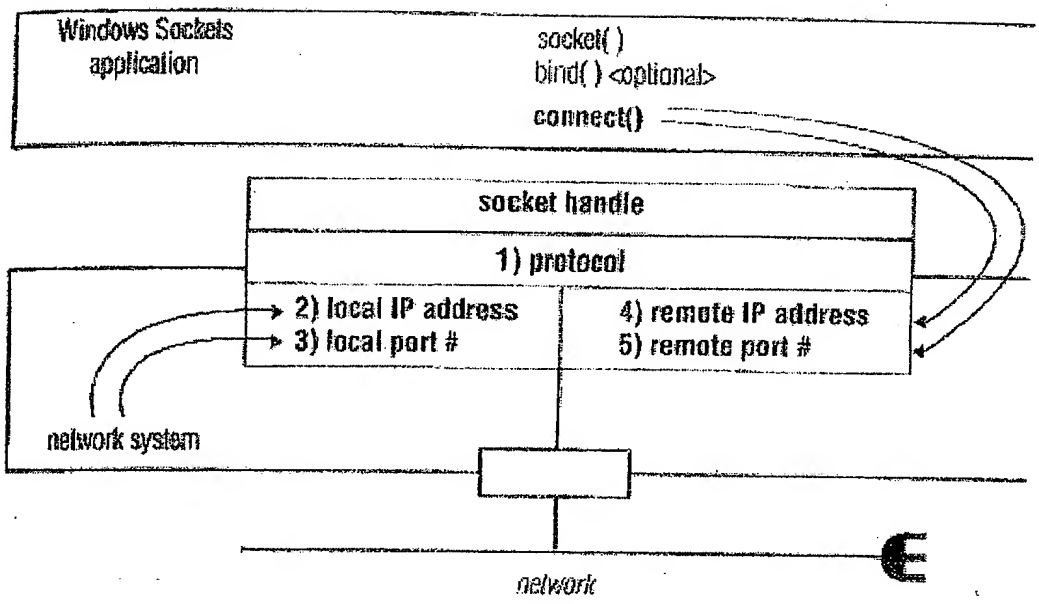

Windows Sockets application
socket( )
bind( ) <optional>
connect()
socket handle
1) protocol
2) local IP address
3) local port #
4) remote IP address
5) remote port #
network system
network

Fig. 12

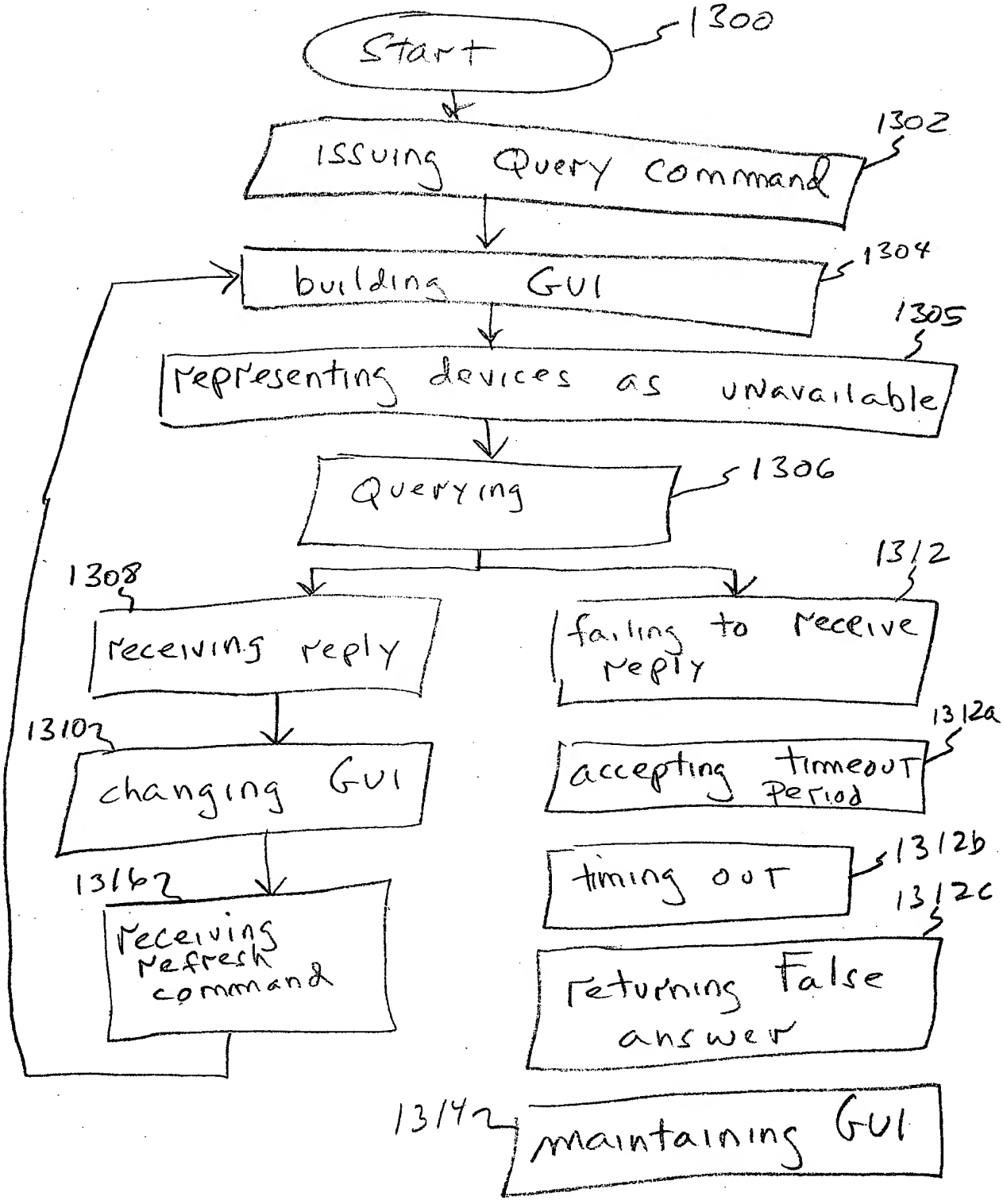

Start
1300
Issuing Query command
1302
building GUI
1304
representing devices as unavailable
1305
Querying
1306
1308
receiving reply
failing to receive reply
1312
13102
changing GUI
accepting timeout period
1312a
13162
receiving refresh command
timing out
1312b
1312c
returning False answer
13142
maintaining GUI

Fig. 13

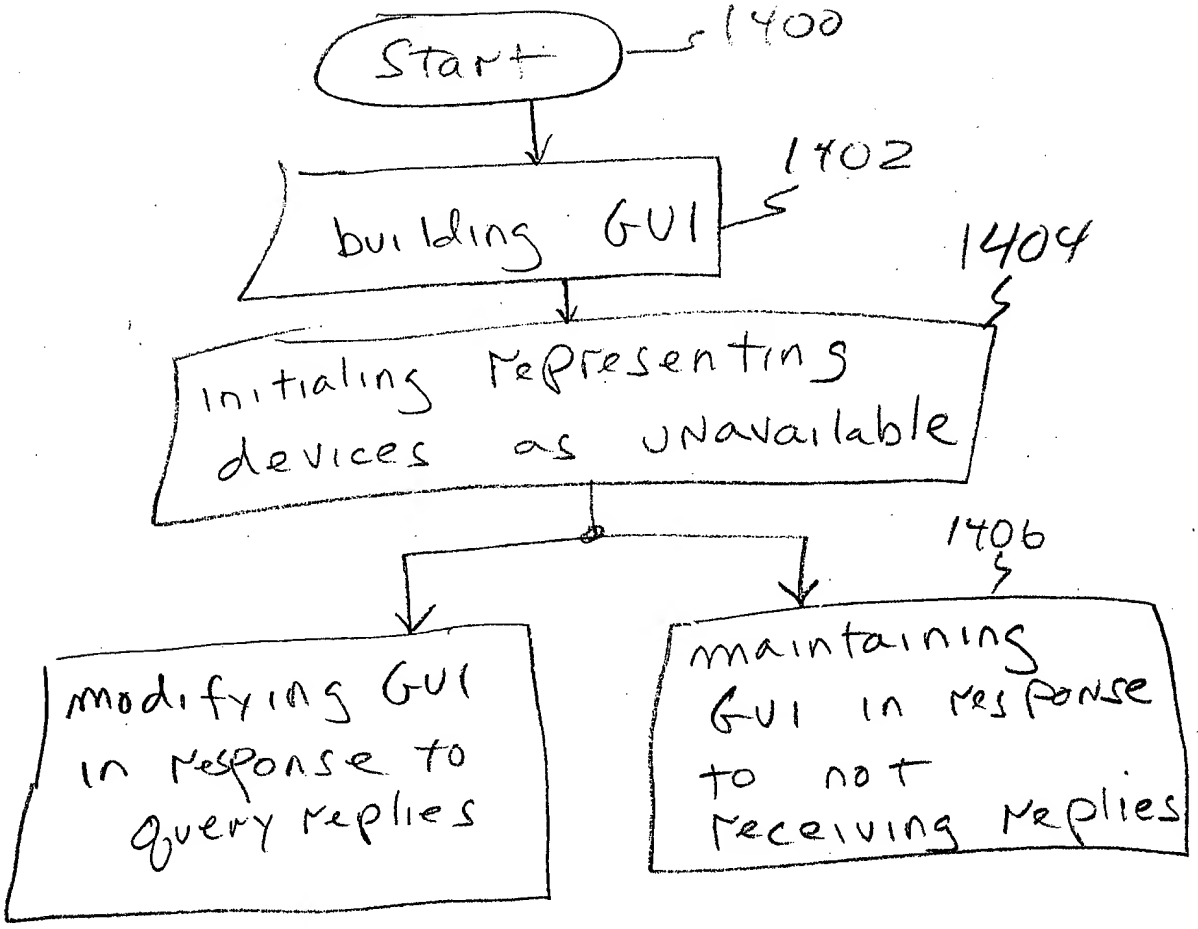

Start
1400
building GUI
1402
initialing representing devices as unavailable
1404
modifying GUI in response to query replies
maintaining GUI in response to not receiving replies
1406

Fig. 14

ATTACHMENT D

Attachment A
In Application Serial No. 09/859,660
Filed May 16, 2001

DECLARATION OF SRIDHAR DATHATHRAYA UNDER 37 CFR §1.132

I, Sridhar Dathathraya, hereby declare as follows:

1. My residence address is 20219, Mapes Ave, Cerritos CA 90703.

2. Since March 2000 I have been employed by Sharp Laboratories of America, Inc. ("SLA"), 5901 Bolsa Blvd, Huntington Beach, CA 92647. My title at SLA, is "Member Technical Staff". My responsibilities include **research and development of application software**.

3. I have read the claims for the patent application in question, System and Method for Discovering Network Components, invented by Guy Eden, Serial Number 09/859,660 (the Applicant). I have read the relevant parts of the Office Action dated September 20, 2004, where claims 2-26 have been rejected as obvious with respect to the Background Section of the Applicant's specification (AAPA), US Patent 5,987,535 (Knodt), and US Patent 6,393,478 (Bahlmann). In summary, it is my opinion that the cited references do not make obvious claims 2-26.

4. The Applicant's Background Section (AAPA) describes a conventional network where the graphical user interface (GUI) is not built until a query is made to all the network devices, and the query responses received. The AAPA clearly describes the time-out problem that can occur if

a query response is not received from a device thought to be available. The inventions described in claims 1, 13, and 15, are a solution to this problem. The claimed inventions build the GUI before the queries are sent.

5. Knodt describes, in greater detail, a system similar to the AAPA. The novelty of Knodt's invention, if any, appears to be in the graphical representation of system components. Channels and component capabilities are depicted on a screen. In many places in the patent (i.e., col. 4, ln. 4-32), Knodt generally touts the dynamic nature of the system, where the screen display provides immediate status. In my opinion, these statements mean that the screen display is responsive to devices inputs (query/response). The only specific example given by Knodt (Fig. 14), shows the screen display being updated in response to discovery (query/response). For example, in Fig. 14, the display features are dampened (Step 84) in response to a query that determines that a device is not in use (Step 82). Other steps in the figure show the same relationship between queries and screen updates. Like the AAPA, Knodt fails to describe a discovery mechanism that builds a GUI prior to initiating a query to a network device.

6. In my opinion, the combination of Knodt and the AAPA do not make the inventions of claims 1, 13, and 15 obvious. More specifically, I have been asked to consider whether Knodt suggests a modification to the AAPA that would make the claimed inventions obvious. I see no comments or drawings in the Knodt application that would suggest to me, or any other skilled artisan, that the AAPA conventional discovery mechanism can be modified. That is, I do not see how the Knodt reference would suggest that a GUI be built prior to the discovery process. As I mentioned above, Knodt is very clear is describing processes that only update the screen after device queries/responses.

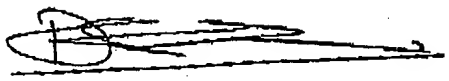
7. Bahlmann describes a network troubleshooting tool. Most of the subject matter of the Bahlmann application is outside the scope of the present discussion, and the reference is mentioned in the Office Action to introduce the subject of the NSLookup function. However, if the element of the NSLookup function is added to the AAPA or to Knodt, this combination of references still does not make the invention of claim 1 obvious. As I mentioned above, Knodt makes no suggestion that the AAPA be modified into a method that builds a GUI prior to discovery. The addition of the NSLookup function to Knodt, still makes no suggestion that the AAPA be modified so as to make claim 1 obvious.

8. In summary, the combination of the Knodt and AAPA do not make the inventions of claims 1, 13, and 15 obvious, since they do not suggest that the GUI be built prior to discovery. Therefore, all the claims dependent from claims 1, 13, and 15 are non-obvious. Likewise, the combination of Bahlmann, Knodt, and the AAPA do not make the invention of claim 1 obvious for exactly the same reason; they do not suggest a modification to the AAPA that builds the GUI before discovery. Therefore, claim 8, which is dependent from claim 1, cannot be obvious.

9. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements were made with the knowledge that willful, false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United State Code and that such willful, false statements may jeopardize the validity of the application or any patent issuing thereon.

10-13-2004

Date



Sridhar Dathathraya